

Kiirloeng

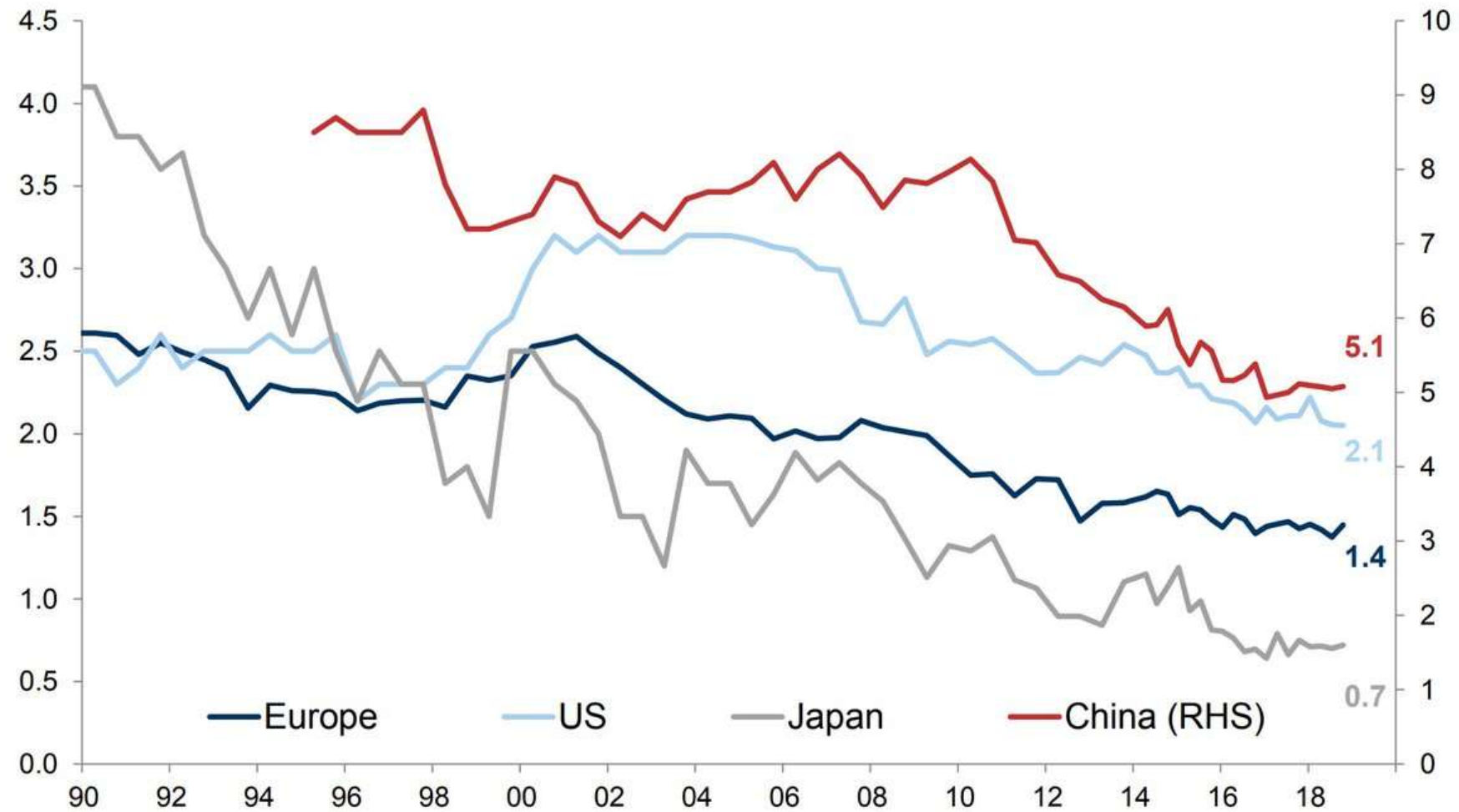
IT ülevaade:

**mis IT-s üldse tehakse +  
väike valik teemadest**

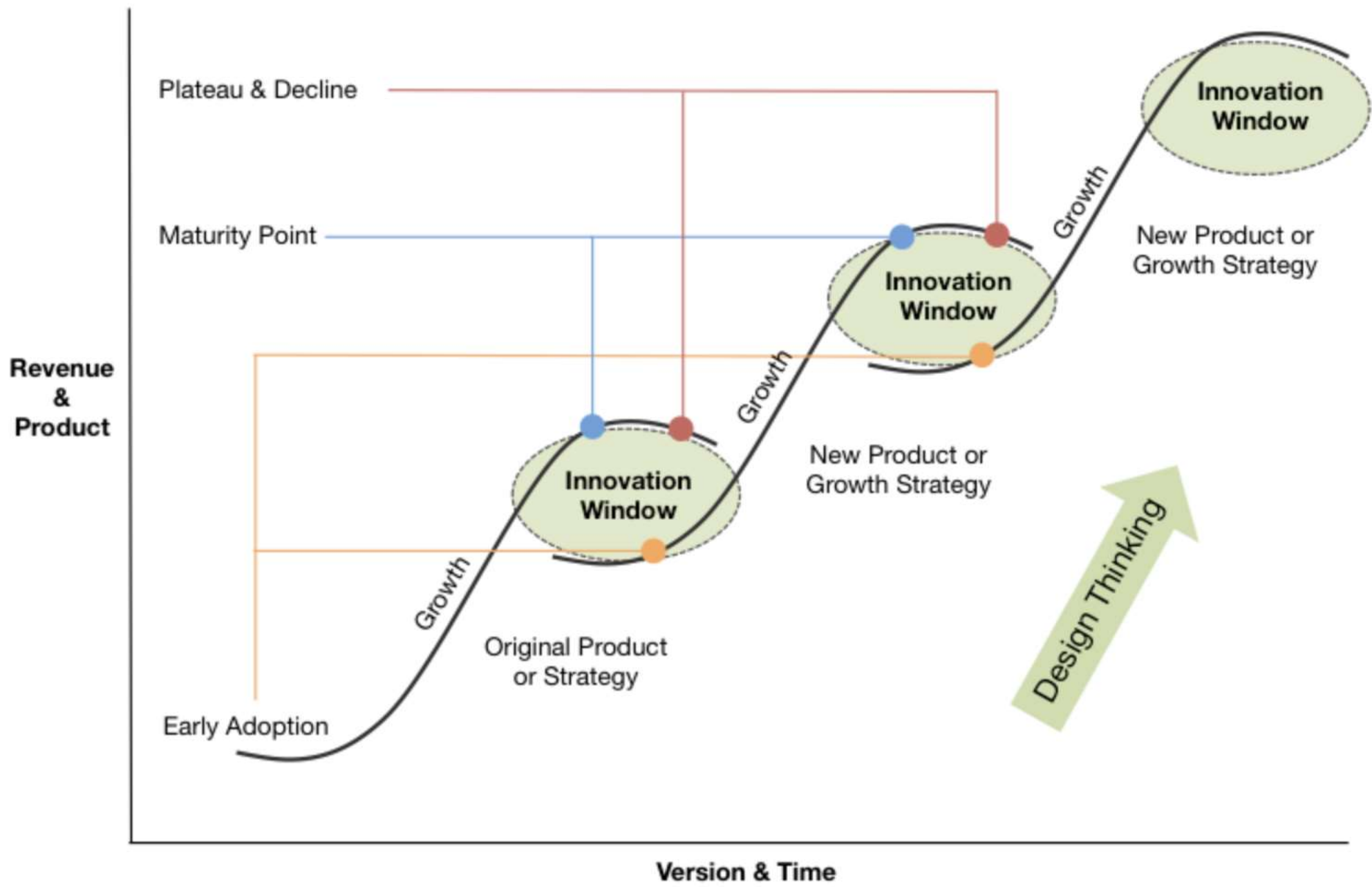
Tanel Tammet

# Exhibit 8: Long-term real global GDP growth forecast is at a historical low

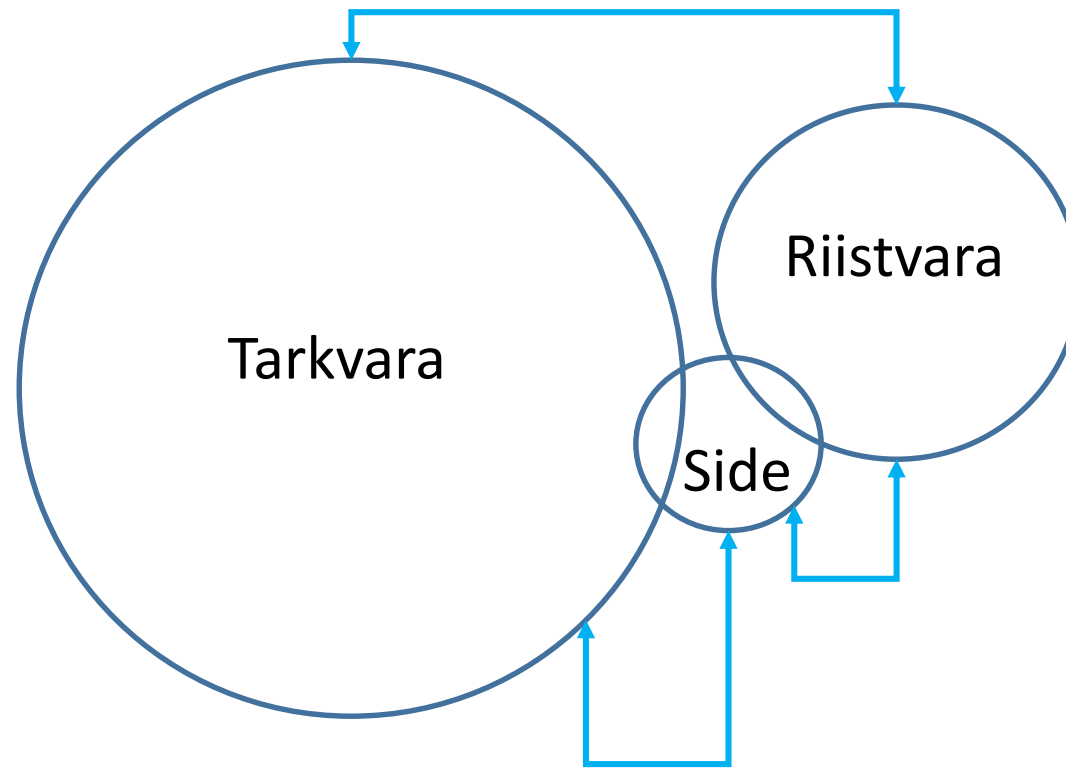
Long-term (6-10y) GDP growth from Consensus Economics



Source: Consensus Economics, Goldman Sachs Global Investment Research

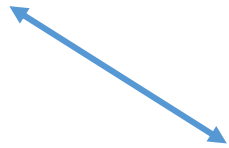


*Üks valdkondlik vaade:*

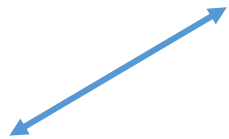


*Teine valdkondlik vaade:*

Arvutiteadus (*teha kõike võimalikult keeruliselt*)



Tehnoloogia / inseneeria (*teha kõike võimalikult lihtsalt*)



Äri / rakendused (*teha klientidele meeldivaid asju*)

# *Kitsam valdkondlik vaade: arvutiteaduse kihid*

## Matemaatiline arvutiteadus

- Algebra eriharud
- Arvutatavus
- Keerukus
- ...

## Sotsiaalmajanduslik arvutiteadus

- E-riik
- Tarkvara analüüs
- Identiteedihaldus
- Küberkaitse organiseerimine
- ...

## Praktilisem arvutiteadus

- Algoritmika
- Verifitseerimine
- Tehisintellekt
- Keeled ja kompilaatorid
- Krüptograafia & küberkaitse
- ...

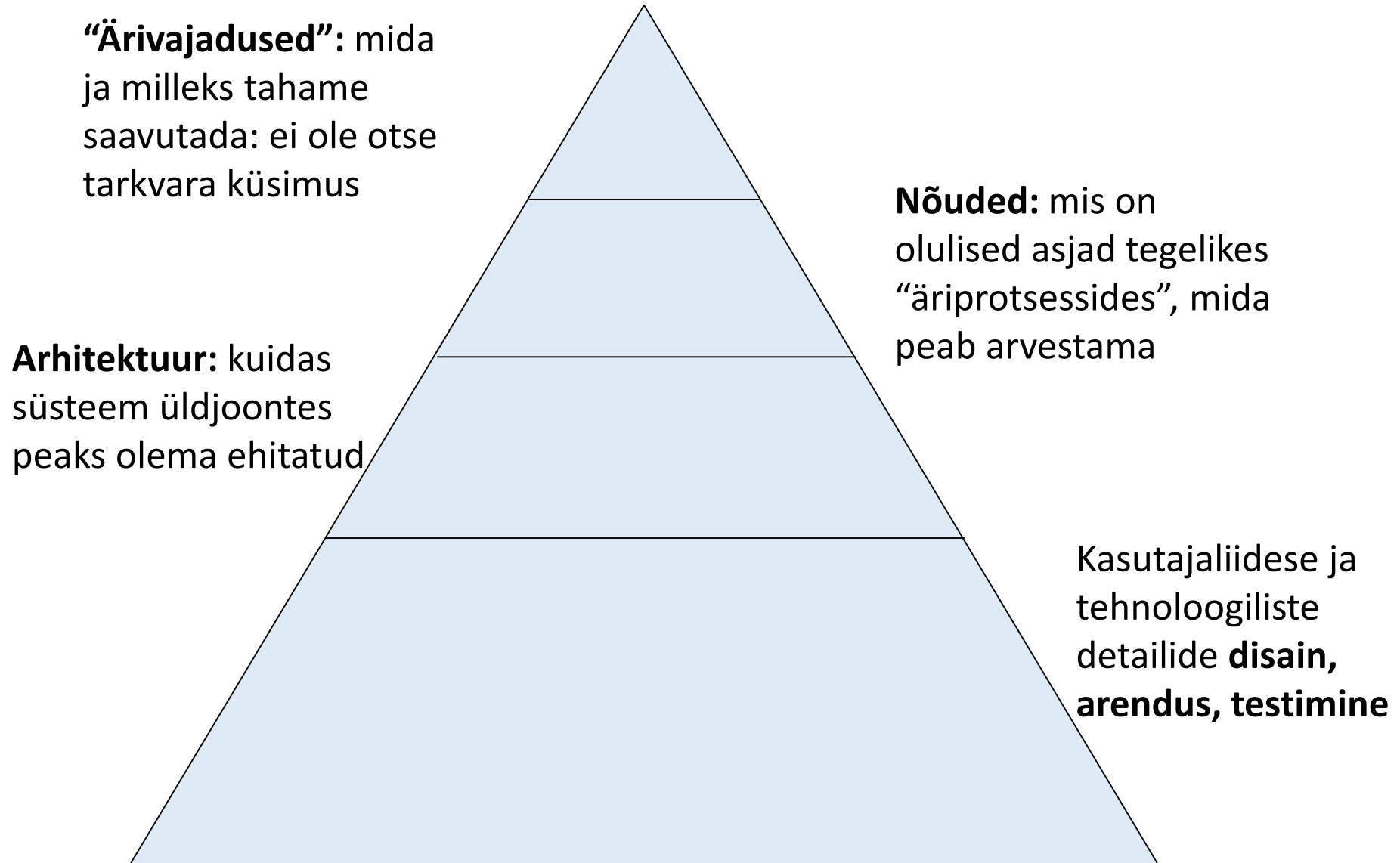
## Veel praktilisem arvutiteadus

- Andmebaaside teooria ja tehnoloogia
- Failisüsteemide ...
- Arvutigraafika ...
- Võrgusüsteemide ...
- ...

## *Tarkvara / ettevõtete põhitüübid klientide ja rahateenimise järgi*

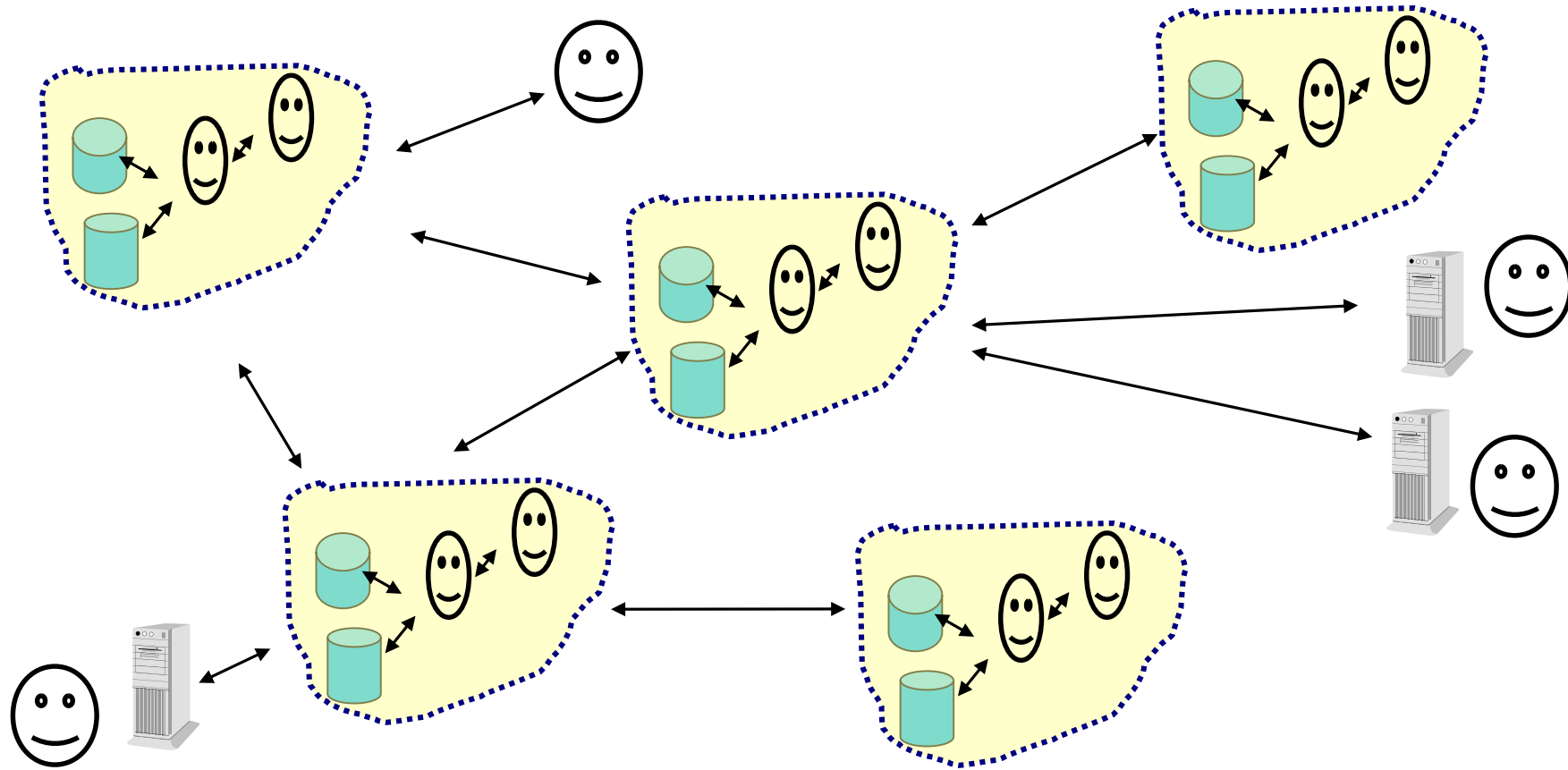
- Eksperimentaaltarkvara: ei ole mõeldud kasutajate jaoks
- Hobitarkvara: kasutajate jaoks, aga mitte raha teenimiseks
- Tootefirmad: teenivad oma tootelt / selle abil
- Agentuurid: teevad tarkvara kliendi tellimusel

# *Suurem pilt: milleks me süsteemi ehitame?*





*Tarkvara on organisatsioonide osa ning suhtleb eri osapooltega*



Tarkvara ehitus ja arendamine

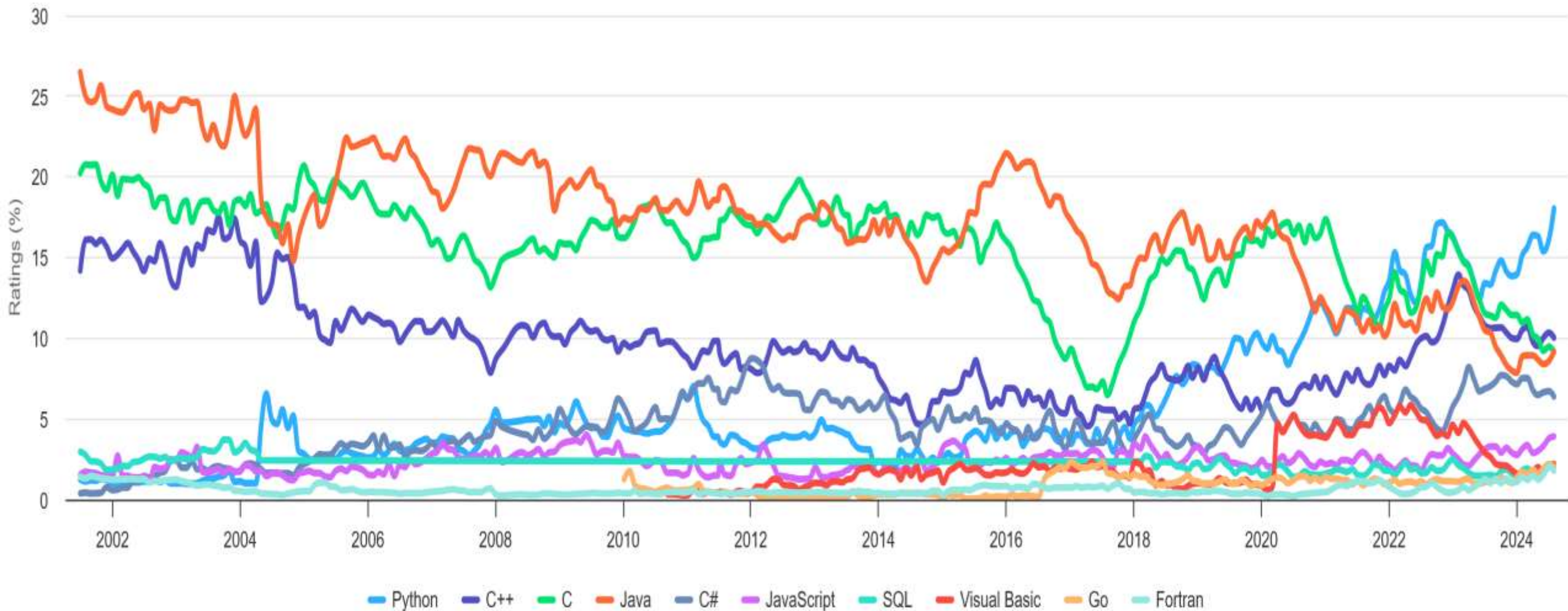
*Mismoodi tarkvara tegelikult ehitatakse?*

Lihtsalt mingis programmeerimiskeeles programme kirjutades.

Keeli on väga palju (tuhandeid), aga populaarseid üpris vähe.

# TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



Väike näiteprogramm *C keeles*: summeerime arve 1 ... N

```
int sumto(int n) {  
    int i, sum = 0;  
    for(i=0; i<=n; i=i+1) {  
        sum = sum + i;  
    }  
    return sum;  
}
```

*Kompileeritud masinkoodi lähedasse assemblerisse **Inteli** jaoks*

Arvuti suudab täita ainult masinkoodi, mitte C-d või Pythonit jne

```
_sumto:
    pushl %ebp                ; Loome ''framepointer''-i
    movl  %esp,%ebp          ;
    movl  8(%ebp),%ecx        ; Võta n.
    xorl  %eax,%eax           ; sum = 0
    xorl  %edx,%edx           ; i = 0
    cmpl  %ecx,%eax           ; Kui i>n ...
    jg    L3                  ; ... mine L3
    .align 2
L5:    addl  %edx,%eax          ; sum = sum + i
    incl  %edx                 ; i = i+1
    cmpl  %ecx,%edx           ; Kui i<=n ...
    jle   L5                  ; ... mine L5
L3:    leave                ; Taasta ebp.
    ret                        ; Valmis!
```

*Kompileeritud masinkoodi lähedasse assemblerisse **ARMi** jaoks*

Arvuti suudab täita ainult masinkoodi, mitte C-d või Pythonit jne

sumto:

```
    mov r1, #0           ; sum = 0
```

```
    mov r2, #0           ; i = 0
```

```
    ldr r3, N             ; r3=N
```

loop:

```
    cmp r3, r2            ; eeldame r3=N
```

```
    bgt finish            ; kui n>i, mine finish
```

```
    add r2, r2, #1        ; i = i + 1
```

```
    add r1, r1, r2        ; sum = sum + i
```

```
    B loop                ; uus tsükkel
```

finish:

```
    end
```

```
N      DCD    #1000      ; mälupea N algväärtusega
```

Väike näiteprogramm *Pythoni keeles*: summeerime arve 1 ... N

```
def sumto(n):  
    sum=0  
    for i in range(n+1):  
        sum=sum+i  
    return sum
```



*Kirjelduskeeled: andmete, kujunduse, päringute jne jaoks: ei ole programmid*

**Json:** andmete esitamiseks

```
{"name": "John Smith", "age": 35, "address": {"city": "Tallinn", "street": "Mere"}}
```

**Html:** veebilehe visuaali kujundamiseks

```
<h1>Pealkiri</h1><p>üks <b>bold sõna</b></p>
```

**SQL:** andmebaasi-päringuteks

```
select name, age from persons where age>18;
```

## *Komponendid*

Tarkvarasüsteemid ehitatakse reeglina mitmesuguste komponentide kokkupaneku, s.t. kokkuprogrammeerimise teel, või teisiti öeldes: komponente kasutades.

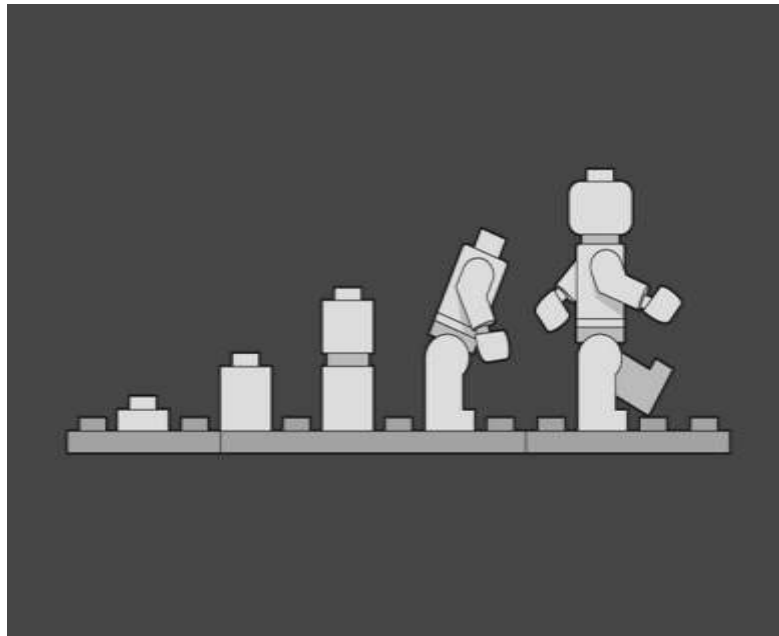
Neid komponente võib klassifitseerida - näiteks - järgmisel viisil:

- Terviklikud lõppkasutaja-rakendusprogrammid, näiteks Excel
- Suured “valmiskomponendid”, näiteks andmebaasimootorid
- Raamistikud ehk frameworks: “pooltooted”
- Teegid ehk libraries: väikeste levinud funktsionaalsuste jaoks (joonistamine, vörk, ...)

## *Tarkvara ei ole Lego*

Tarkvara ei panda kokku “nagu lego blokkidest”

Lego blokke saab ühendada väga lihtsalt, ja neil ei ole sisemist keerulist ehitust

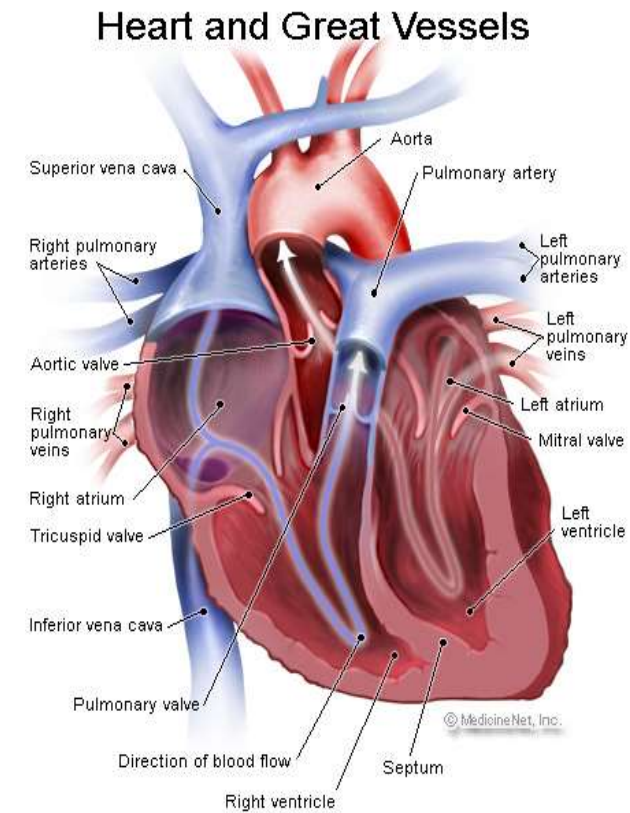


## *Tarkvara on sarnane bioloogiaga*

Tarkvara ei panda kokku “nagu lego blokkidest”: ta sarnaneb enim bioloogiaga

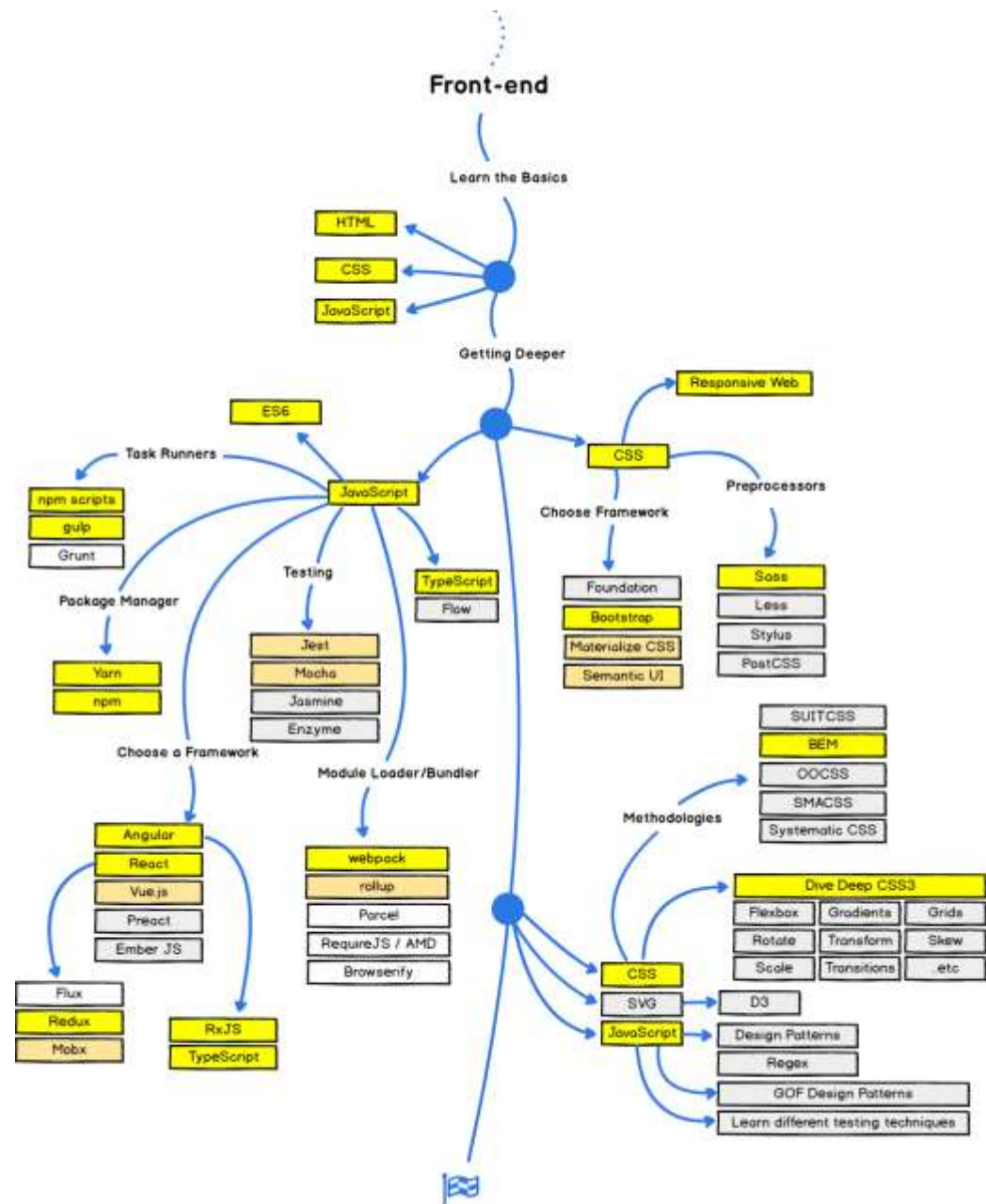
“Tarkvarablokid” on väga keerulised süsteemid, ja nende ühendamine tähendab nõ tavapärast programmeerimist

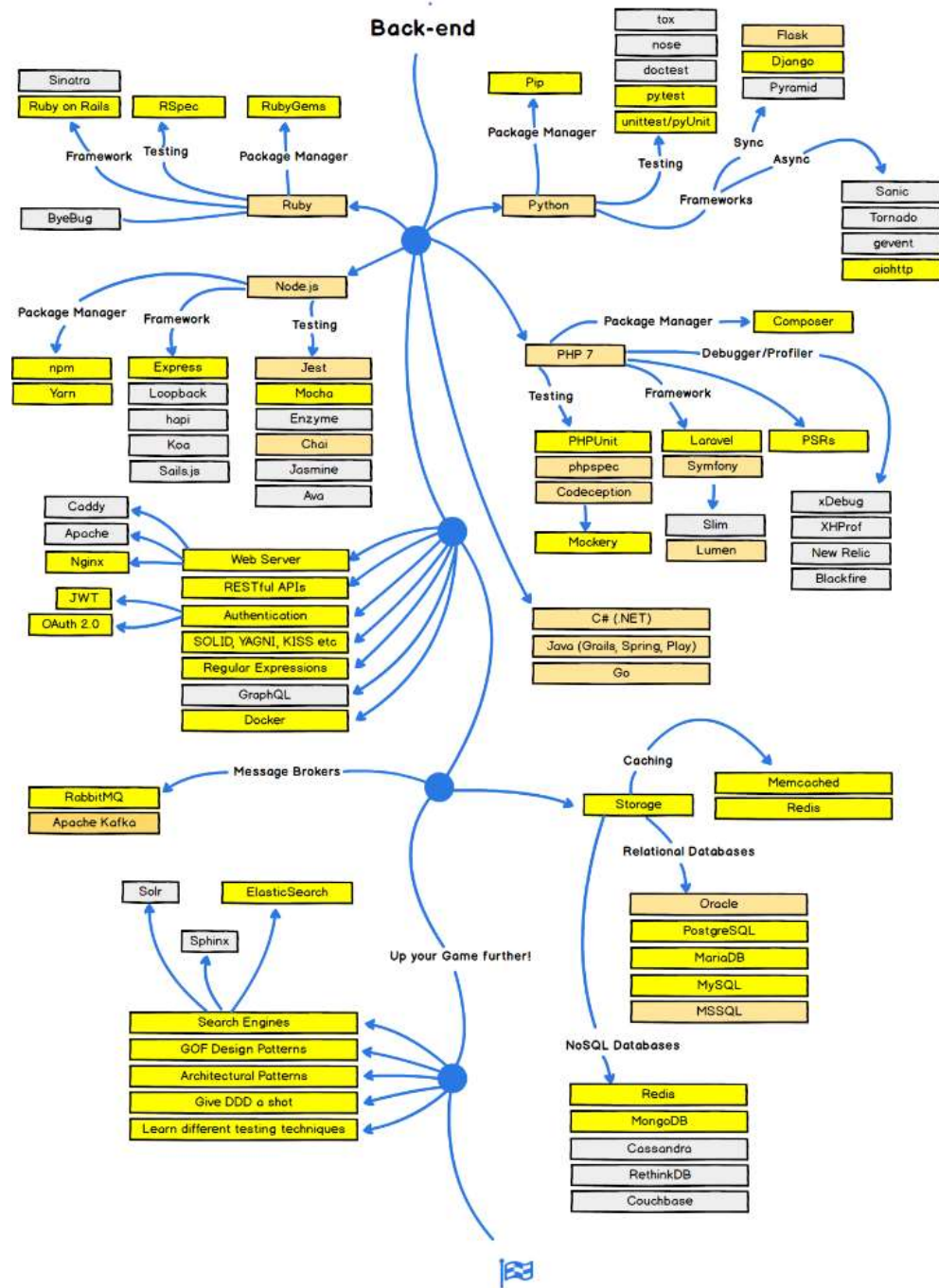
Tarkvara areneb olemasolevast tarkvarast arendajate abil evolutsiooni teel, sarnaselt bioloogiaga



## *Arendustehnoloogiad*

- Programmide teisendamine masinkoodi, interpreteerimine, kokkuühendamine, pakendamine, modifitseerimine.
- Spetsialiseeritud tekstiredaktorid, visualiseerijad.
- Tarkvara muutuste haldajad.
- ... jne jne ...

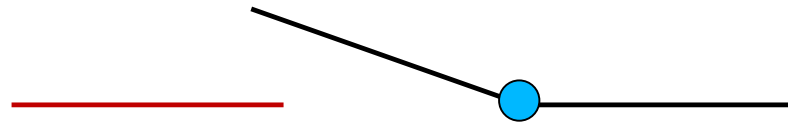




Riistvarast



# Näpuga lüliti

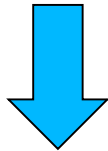


Vool läbi ei lähe

Elekter  
siit  
sisse

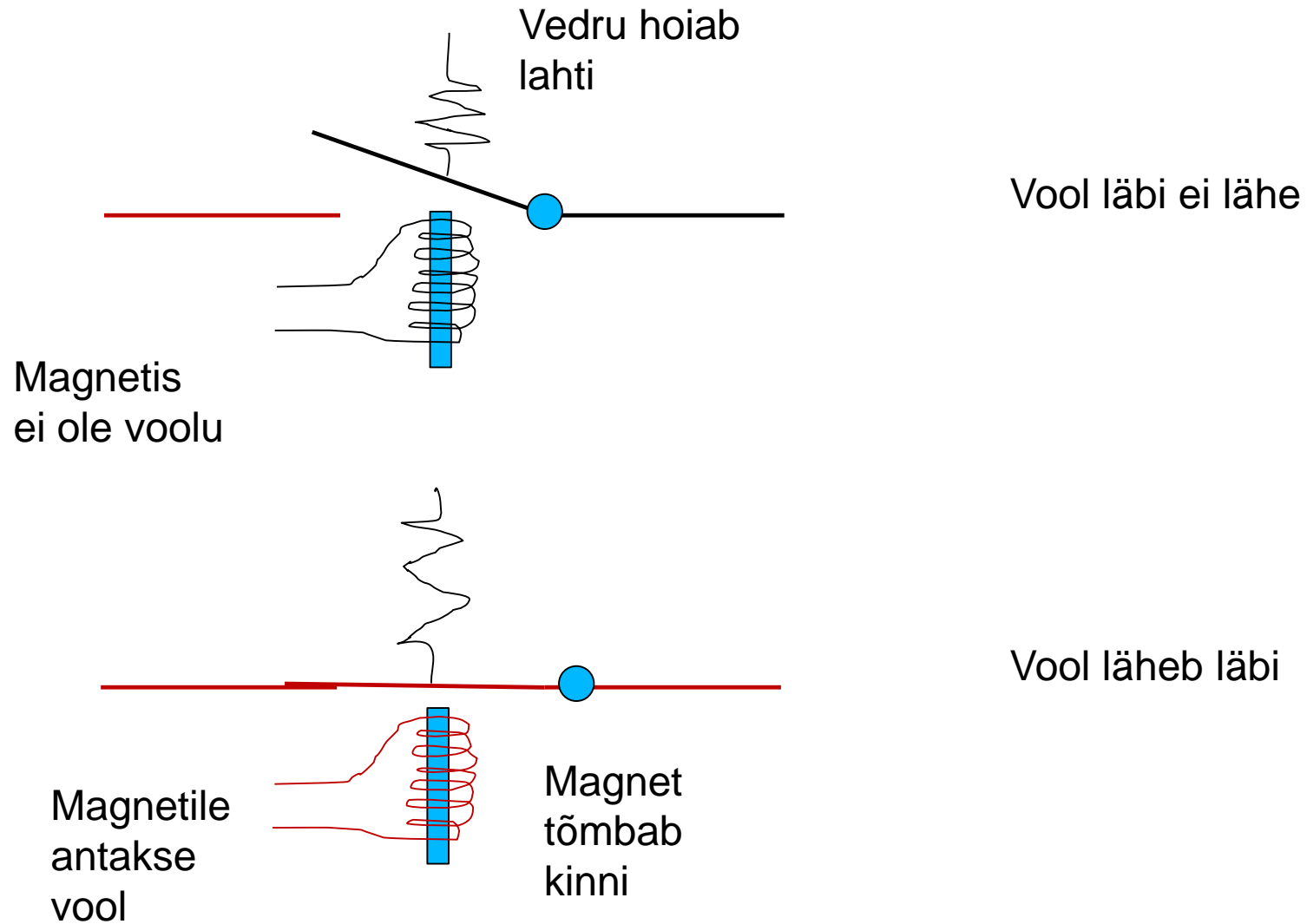
Vajuta  
näpuga

Elekter  
siit  
välja?



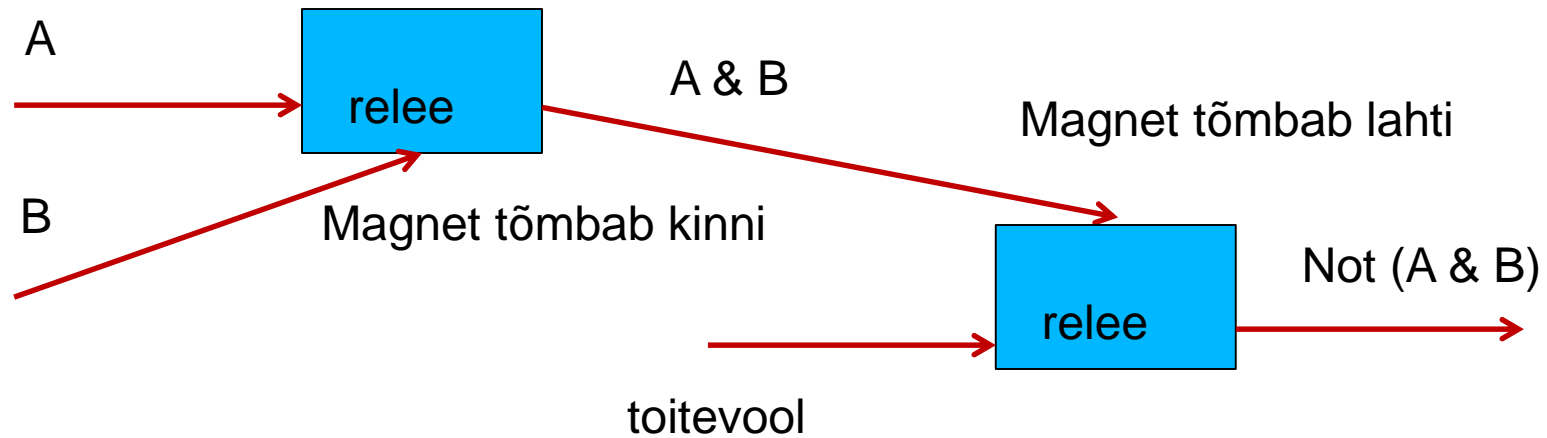
Vool läheb läbi

# Relee: mootoriga lüliti



# Miks releed? Näide.

Realiseerime lausearvutuse valemi **Not (A & B)** releedega:

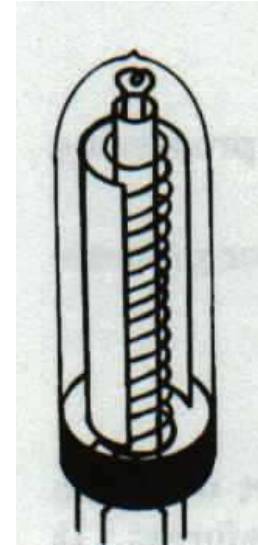


Keerukamaid asju, näiteks liitmist ja korrutamist, ehitatakse suure hulga lausearvutuse tehetena umbes samal moel.

# Raadiolambi tööpõhimõte

- [Vacuum Tube \(1906, Lee Deforest\)](#)

Three elements device used as electronic switch and amplifier: two electrodes separated by a grid in a vacuum glass enclosure.

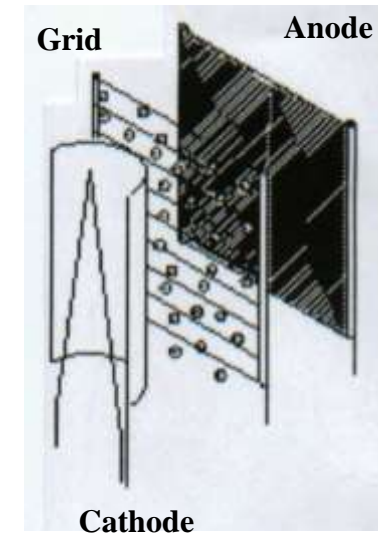


- [Principle](#)

*Cathode* - emits electrons;

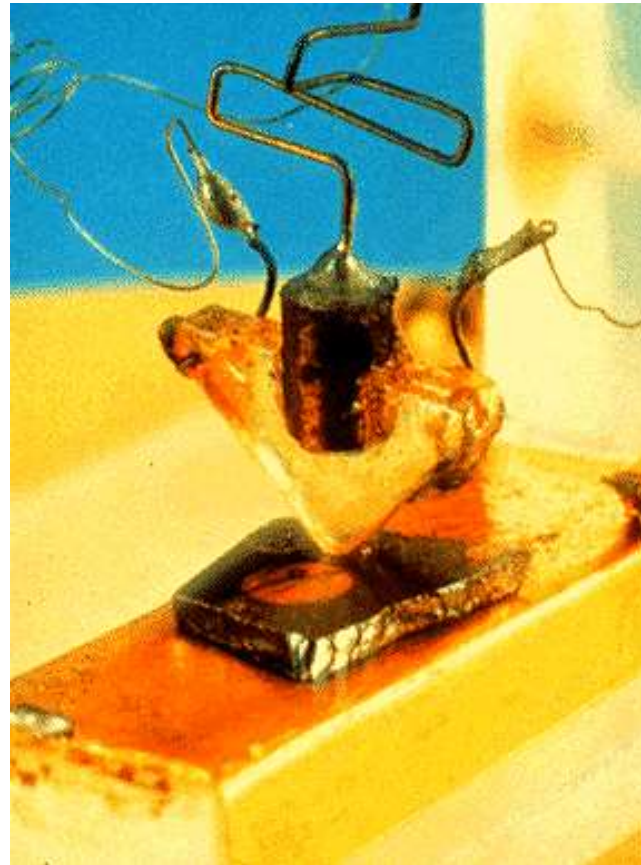
*Plate (anode)* - receives the electrons;

*Grid* - with negative bias voltage repels some of the electrons and prevents them from reaching the plate, resulting in less current flow. A changing negative charge on the grid modulates the plate current.



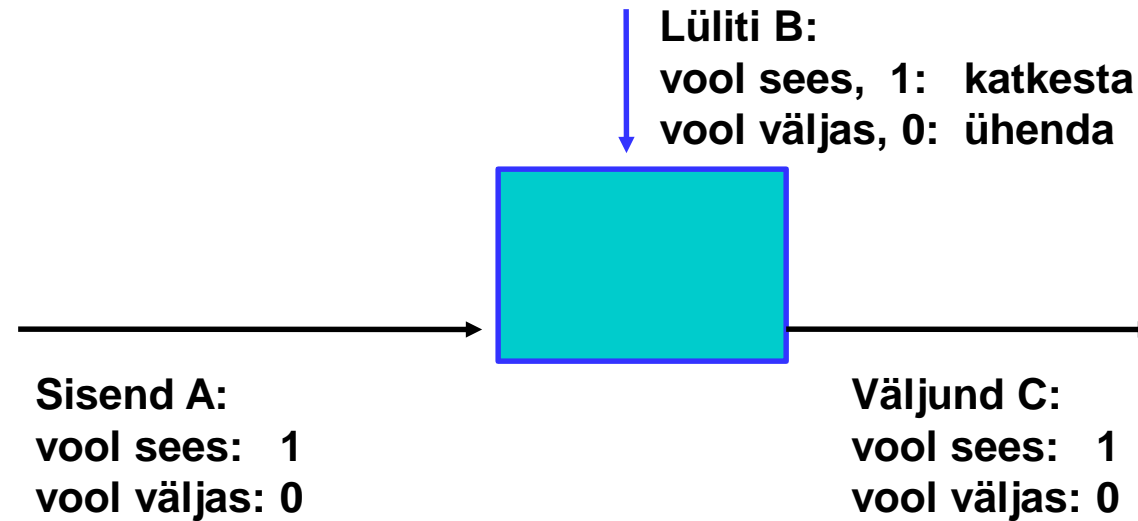
# 1947

Three scientists at Bell Telephone Laboratories, William Shockley, Walter Brattain, and John Bardeen demonstrate their new invention of the point-contact transistor amplifier.



# Komponendid

- Peamine idee: transistorid kui “katkestusmootoriga” lülitid

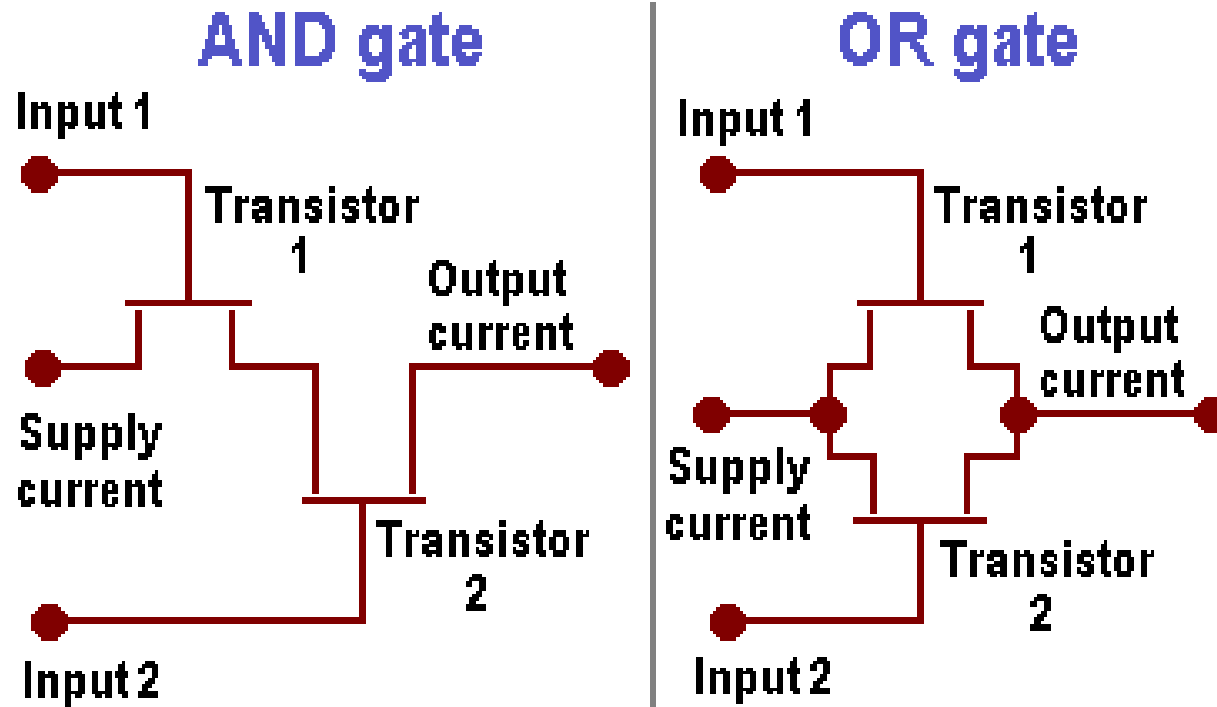


$$C = (A \text{ and } -B)$$

- Väikestest komponentidest ehitatakse suuremaid, millest omakorda veel suuremaid.
- Komponendid on kui mustad kastid: teame nende väljundit vastava sisendi korral, aga enamasti mitte nende tehnilist sisu.

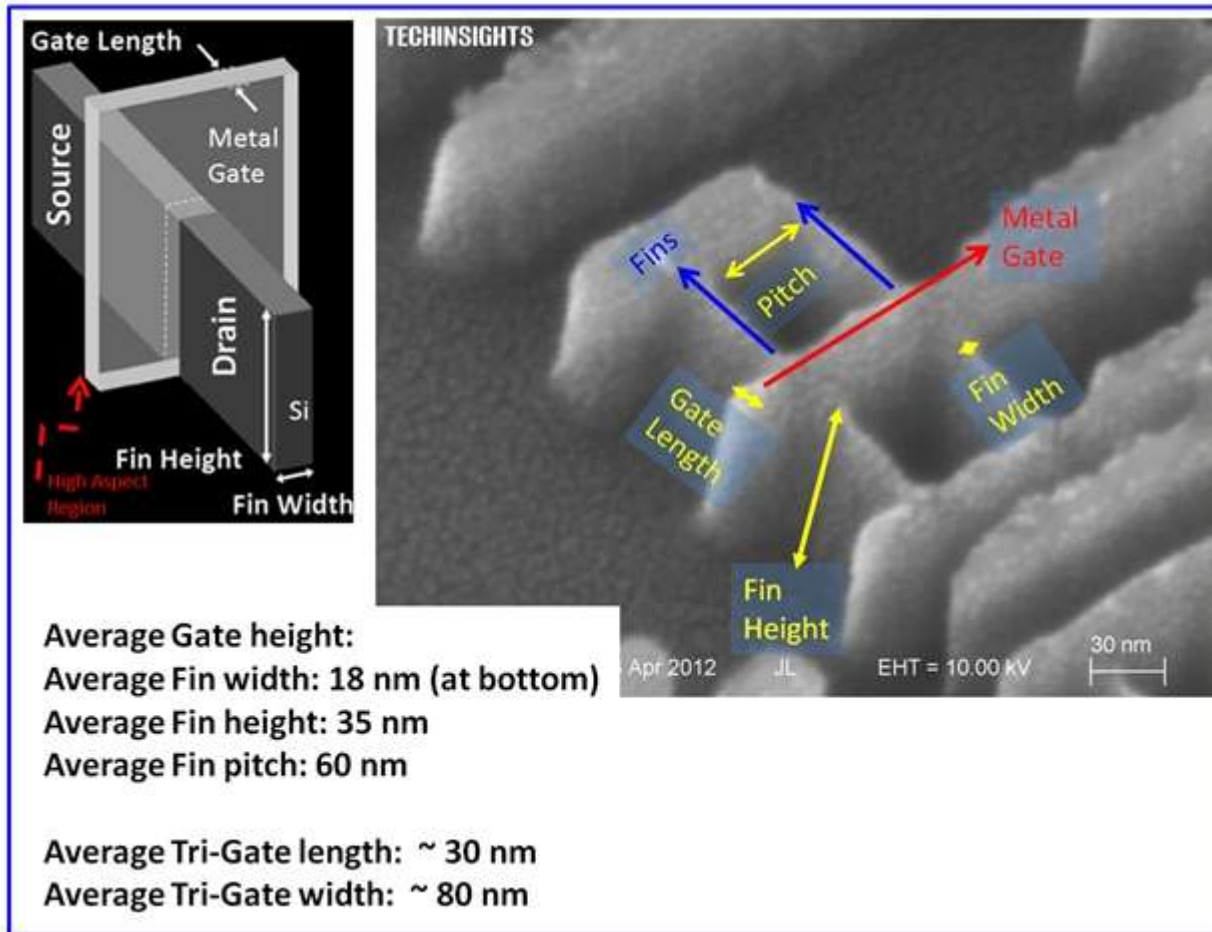
## Logic gates: simple idea

- MOSFET: metal-oxide semiconductor field-effect transistor
- Input near 2V: circuit on. Input near 0V: circuit off.



# Mikroprotessori sees

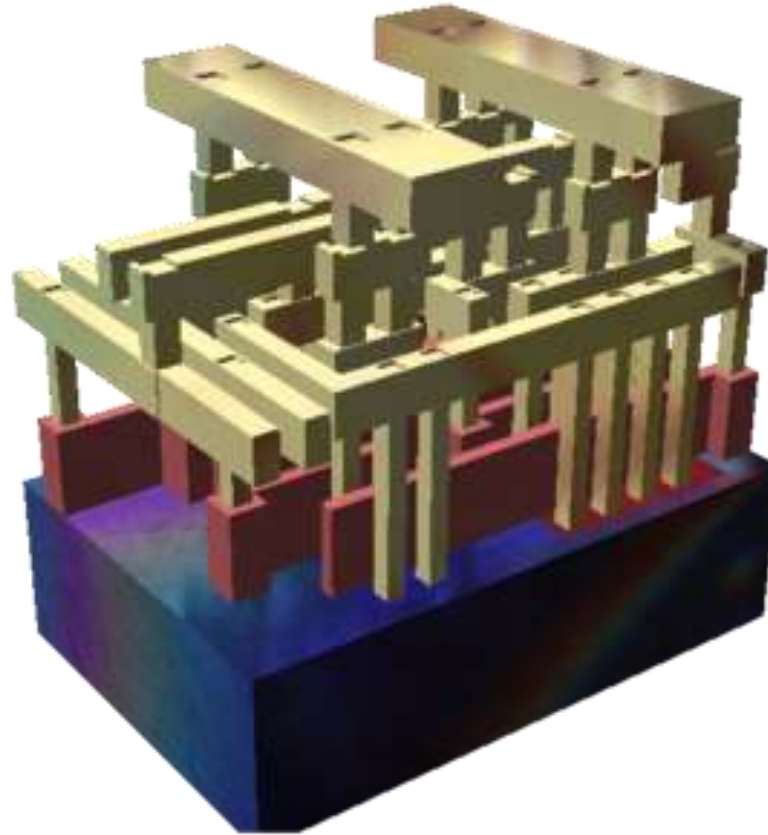
- Photo from Intel CORE i5-3550 processor: Ivy Bridge 22 nm process technology





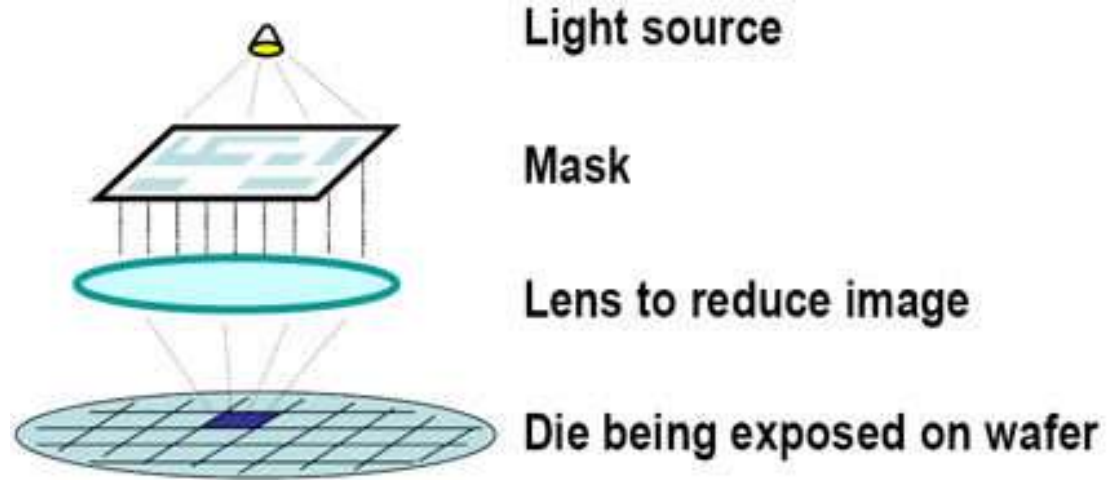
## Layers in an integrated circuit

- Blue: silicon bulk
- Red: polysilicon gates
- Yellow: metal wires



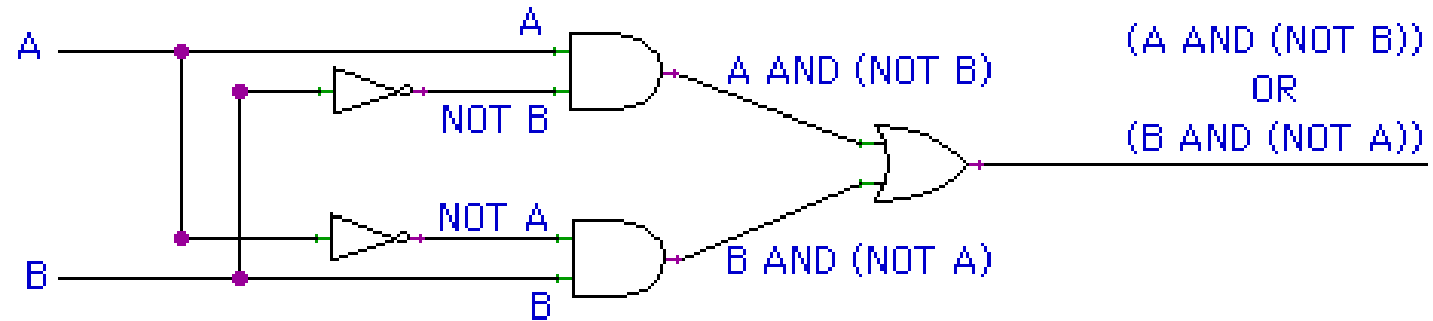
# Photolithography

- First a mask is made for each layer
- The mask is projected to a tiny area on the silicon wafer



# Komponendid

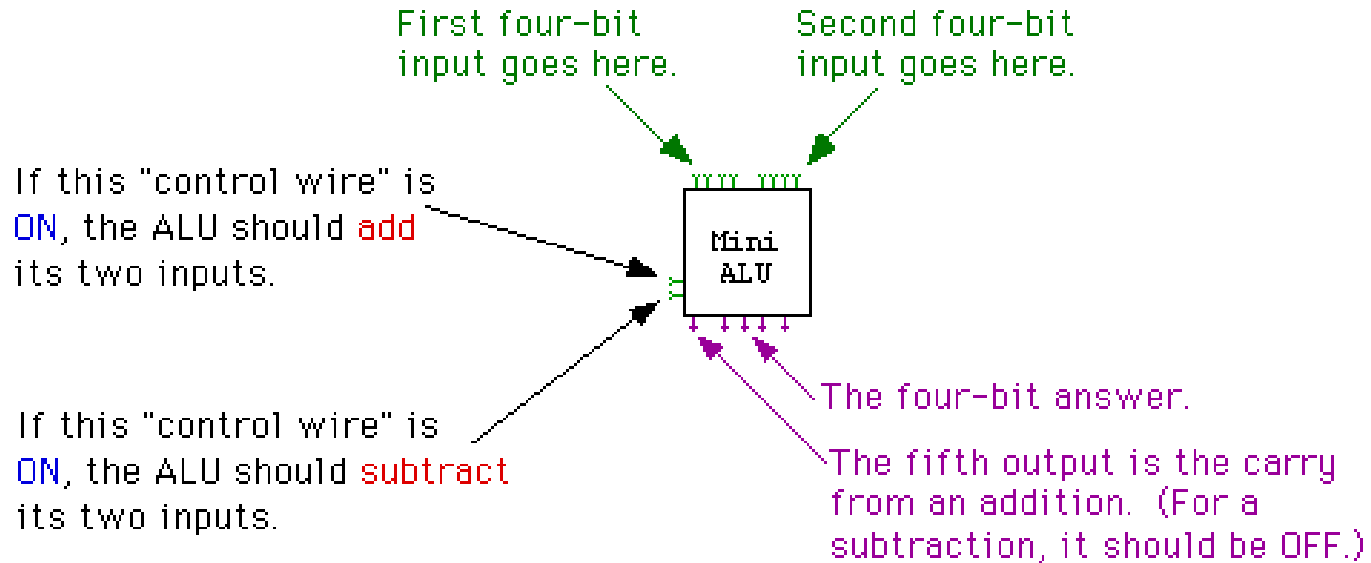
Näiteks: (A and (not B)) or (B and (not A))



## Neljabitine liitja (four-bit adder)

- <https://math.hws.edu/eck/js/xLogicCircuits/xLogicCircuits.html?file=combinational.txt>
- Kaheksa pluss kaks sisendjuhet, neli pluss üks väljundjuhet

1011	1111	1111	1010	0111	0001
0110	0001	1111	0101	1010	0011
-----	-----	-----	-----	-----	-----
10001	10000	11110	01111	10001	00100



Tehisintellektist

## *Moravec'i paradoks:*

Inimeste jaoks rasked ülesanded on masina jaoks tihtipeale lihtsad.

Inimeste jaoks lihtsad ülesanded on masina jaoks tihtipeale äärmiselt keerulised.

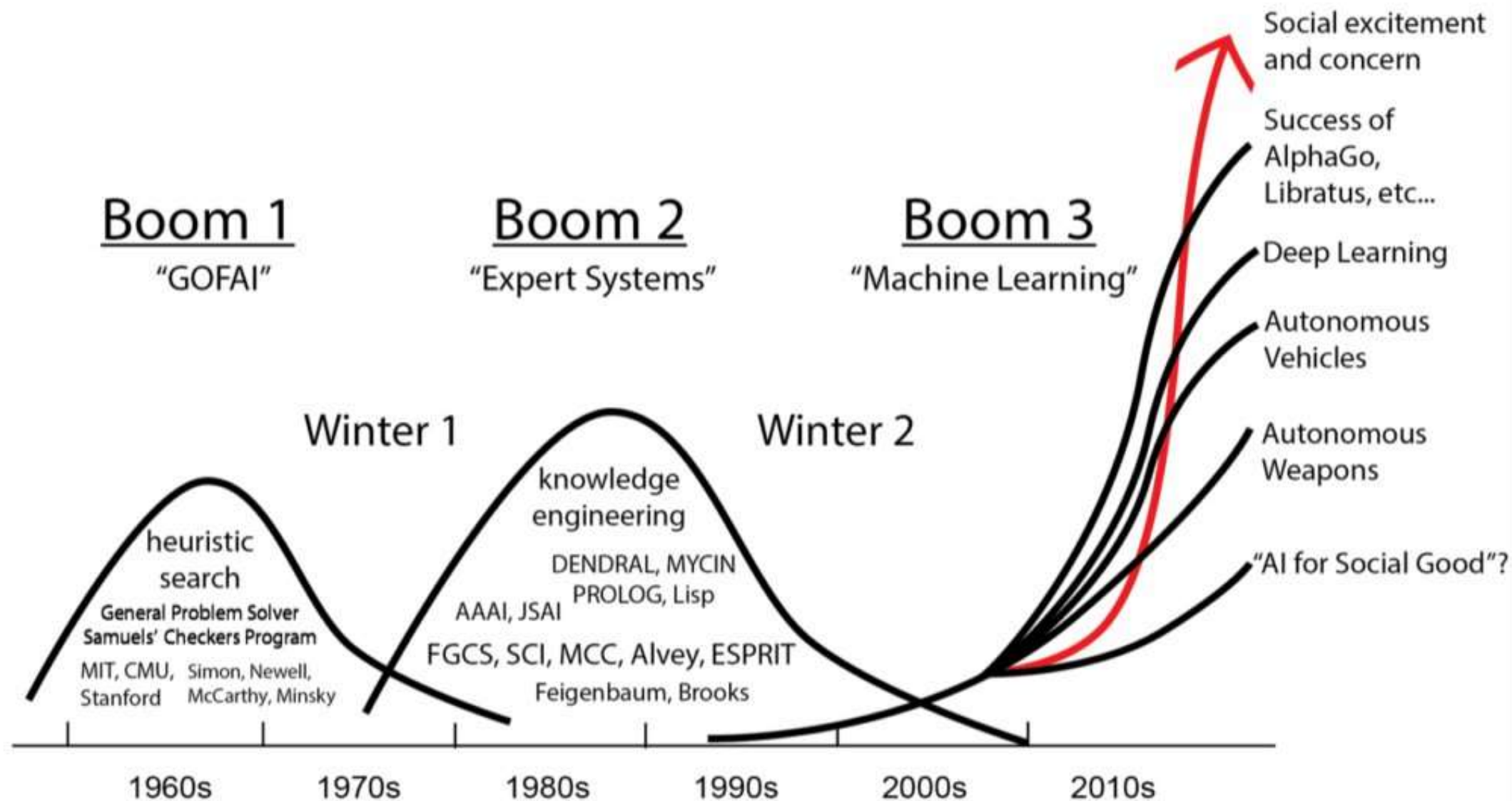
*Amara printsiip:*

Meil on kalduvus lähiperspektiivis tehnoloogia mõju üle hinnata ja pikemas perspektiivis alahinnata.

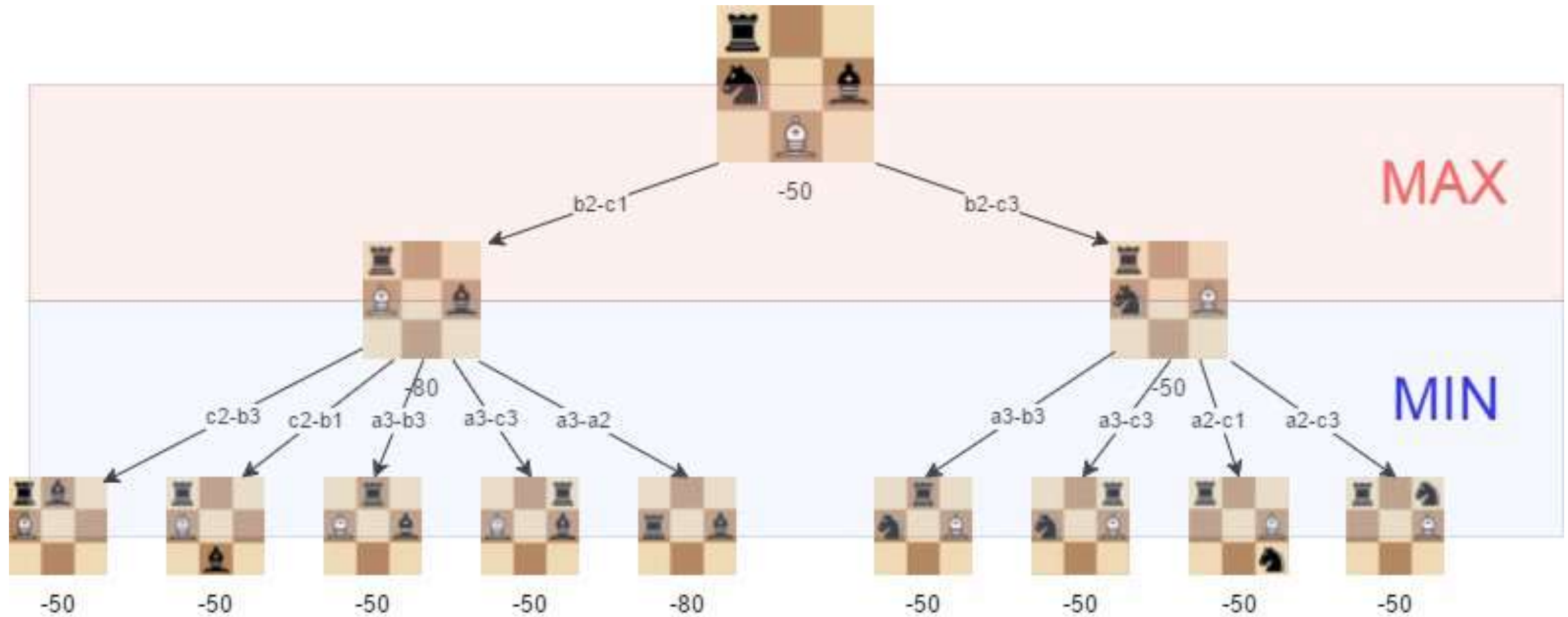
## *Kaks suurt A.I. meetodite valdkonda*

- **Otsing** (sümbol-A.I.)  
paljude variantide seast a la malemäng, matemaatikaõestus, teekonna optimeerimine, planeerimine jne.
- **Masinõpe** (“neuro”-A.I.)  
näidetelt üldistamine, a la objektide äratundmine pildilt, deepfake, aegridadelt ennustamine, suured keelemudelid (LLM) GPT, LLaMA, Claude jne.





# *Otsing variantide hargnevast puust*



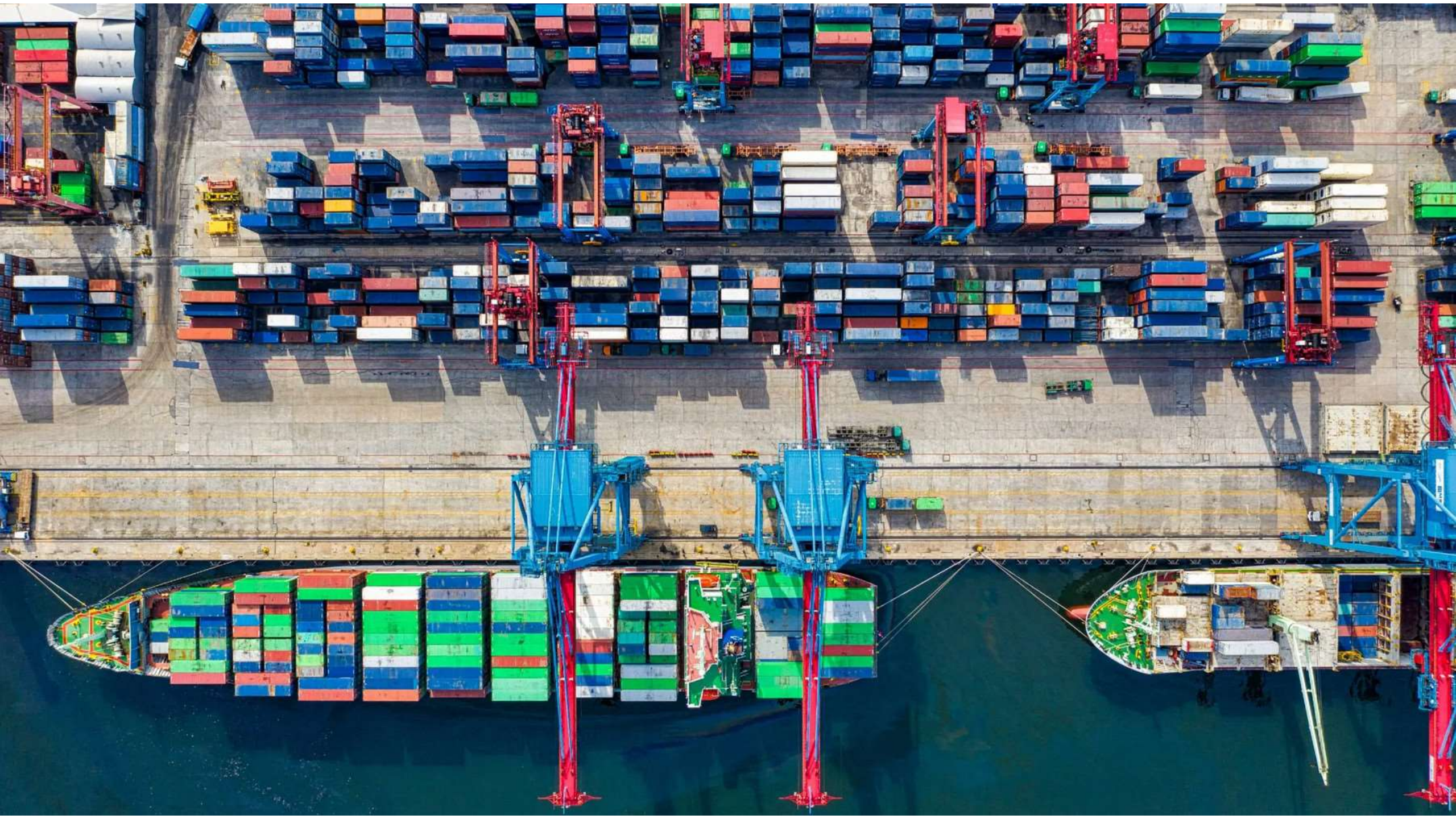
Unoptimized Route



Optimized Route









*The case of logic: a universal method to represent knowledge and derive inevitable conclusions – new knowledge -- from existing knowledge*

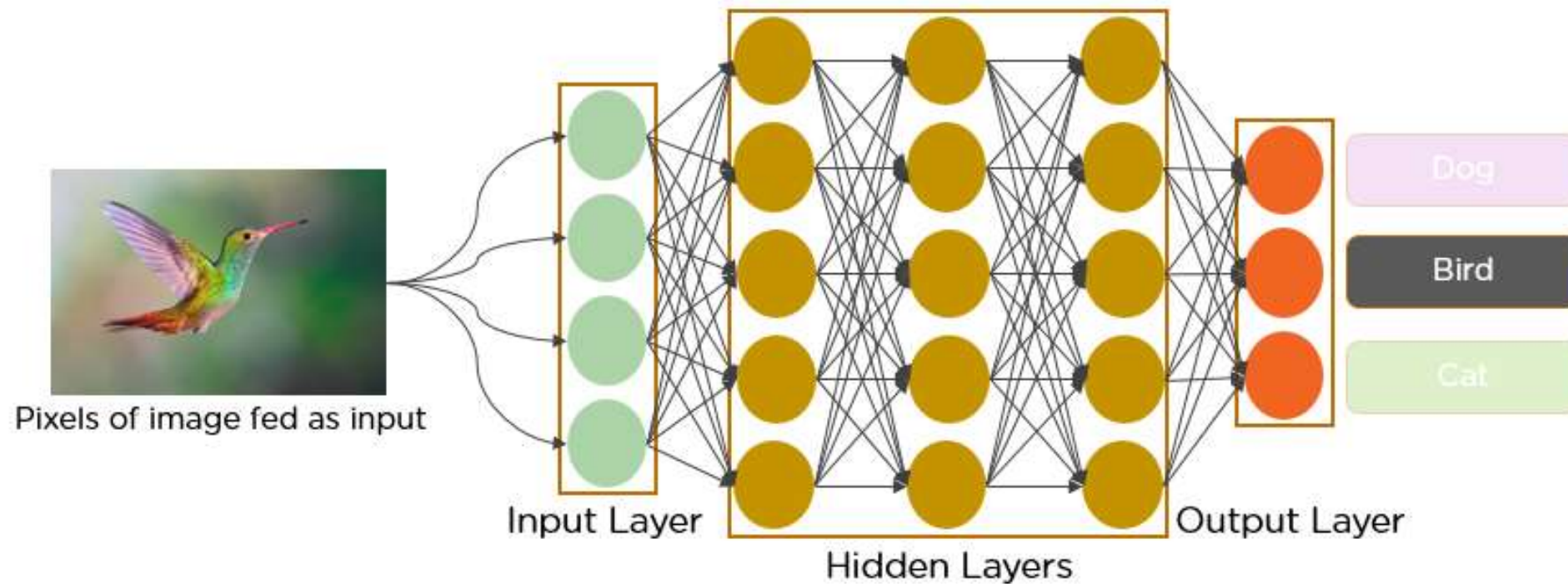
### Issues:

- Formalising knowledge and goals is a very complicated, hard work
- Existing formalisms are not well suited for uncertain knowledge
- Existing knowledge bases cover very little of commonsense knowledge
- Actually finding proofs is extremely hard

## *Masinõpe*

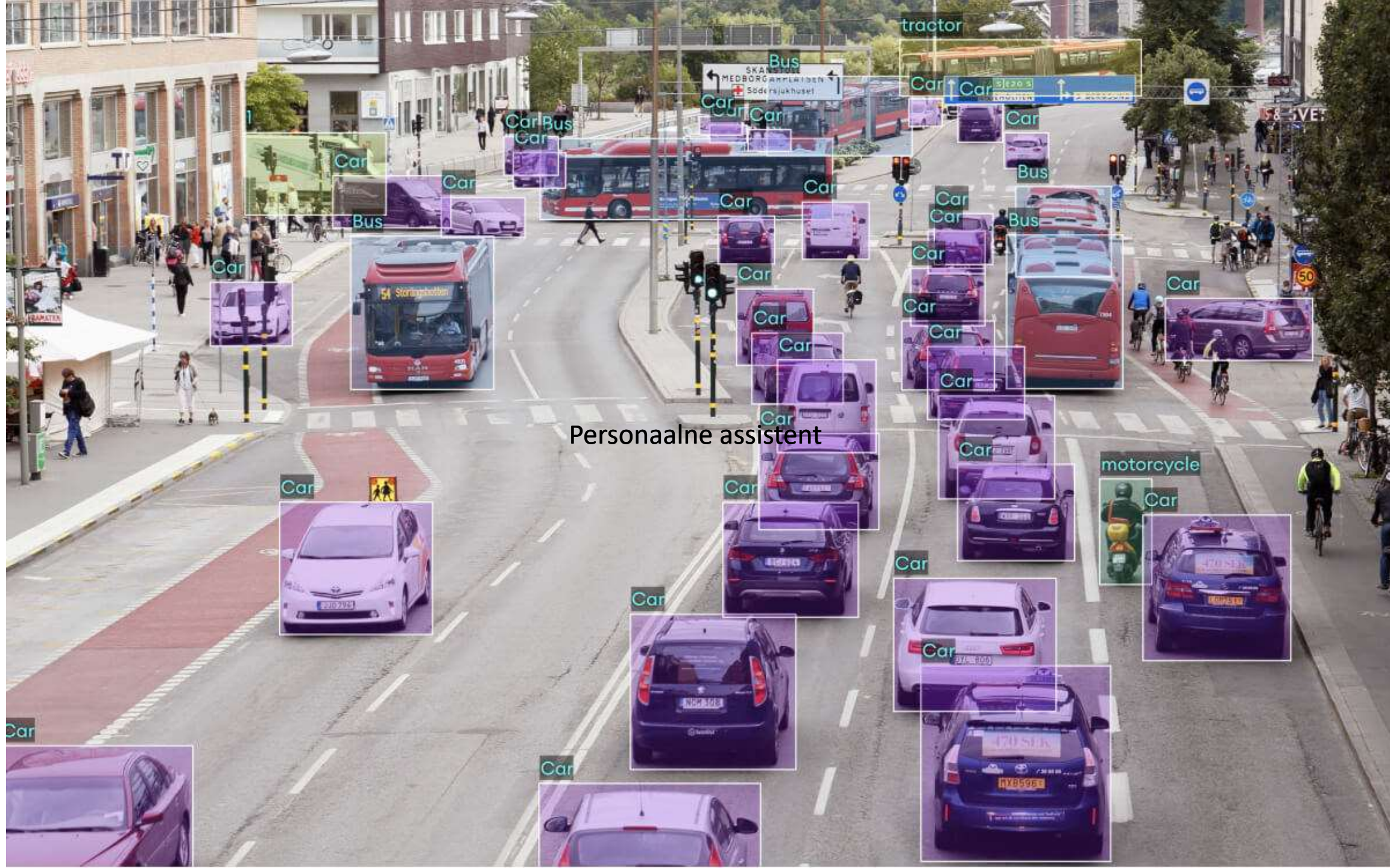
- Lihtsustatult öeldes: “täiustatud statistika”.
- Kõige tuntum meetodite klass on **närvivõrgud**, mis omakorda jagunevad paljudeks alamtüüpideks.
- Kasutusel ka suur hulk **mitte-närvivõrkude**-põhiseid masinõppe algoritme: vektormasinad, lähimad naabrid, juhustemetsad, ...

## *Deep neural networks* ehk sügavad närvivõrgud





Personaalne assistent





(a)



**school bus** 1.0

(b)



**garbage truck** 0.99

(c)



**punching bag** 1.0

(d)



**snowplow** 0.92



**motor scooter** 0.99



**parachute** 1.0



**bobsled** 1.0



**parachute** 0.54



**fire truck** 0.99



**school bus** 0.98



**fireboat** 0.98



**bobsled** 0.79

*Mida suured keelemudelid (LLM) teevad?*

*Ennustavad samm-sammult teksti jätkumist:*

- Jaan istus autosse ja ...
- Jaan istus autosse ja hakkas ...
- Jaan istus autosse ja hakkas sõitma ...
- Jaan istus autosse ja hakkas sõitma kodu ...

## *Kuidas LLM ennustab? Lihtsustatult:*

- Kogub statistikat suure tekstihulga pealt: mismoodi seal on laused jätkunud, näiteks

→ Mihkel istus mootorrattale ja ... (*hakkas, mõtles, ...*)

- Kui näeb lauset, mille kohta statistikat ei ole, kasutab statistikat varasemate sarnaste lausete kohta:

→ Jaan istus autosse ja ...

## *LLMi põhisuutlikkused*

- Üliinimlikul tasemel ahvimine: genereerib sobivas stiilis mõistlikku teksti.
- Head üldteadmised.
- Hea kategoriseerija.
- Suudab kasutada näiteid.
- Suudab teha mõõdukaid üldistusi ja lihtsamaid järeldusi.
- ...

## *LLMi põhiprobleemid*

- Hallutsineerib ehk leiutab “fakte”, mis ei ole üldse tõesed.
- Vaikimisi omab ainult üldteadmisi.
- Ei ole mälu: uusi asju raske juurde õpetada.
- “Must kast” ehk ei anna tegelikke seletusi, kuidas vastused saadi.
- Ei sisalda otsialgoritme ja enamasti ei suuda teha pikemaid järeldusahelaid.
- Ei ole lihtsat viisi siduda andmebaasidega, inseneriarvutustega jne.
- Vastuste kvaliteet langeb kiirelt, kui sisendtekst on pikk.
- Ei ole selget viisi tema suunamiseks või mingit sorti vastuste keelamiseks.
- ...

**Human:** A man and a goat are on one side of the river. They have a boat. How can they both go across?

**ChatGPT:** The man takes the goat across the river first, leaving the boat on the original side. Then, the man returns alone with the boat. He leaves the goat on the other side and takes the boat back to the original side. Finally, he takes the cabbage across the river.

*Neurosymbolic A.I. aka hybrid A.I.: a research area for combining symbolic and ML methods*

**Some notes:**

- An active research area, not an established applications toolkit
- A huge number of different possible approaches
- Existing practical A.I. systems are mostly already neurosymbolic: autonomous driving, leading chess engines, math proof systems, google search, alexa & siri, successes at geometry etc



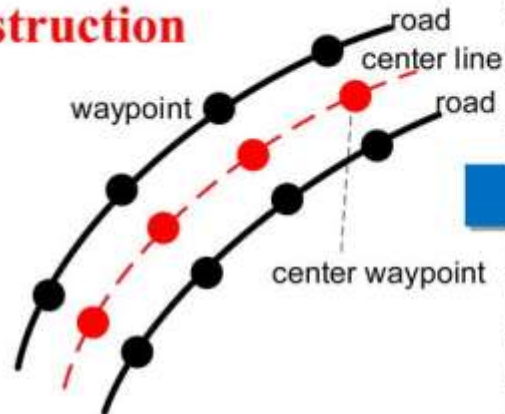
*Hübriidsüsteemid,  
näiteks isesõitvad  
autod*

Pildil: 2005,  
Stanfordi Stanley  
võidab DARPA  
Grand Challenge II

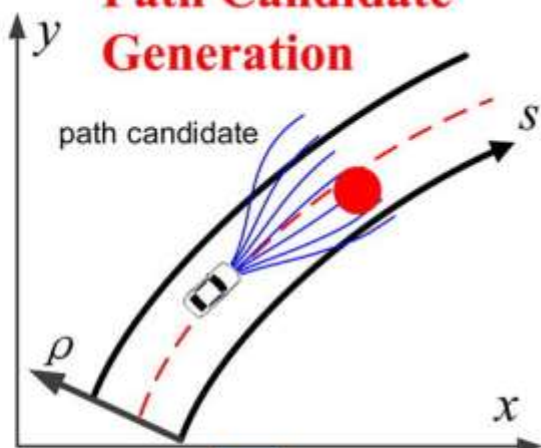




### Center Line Construction

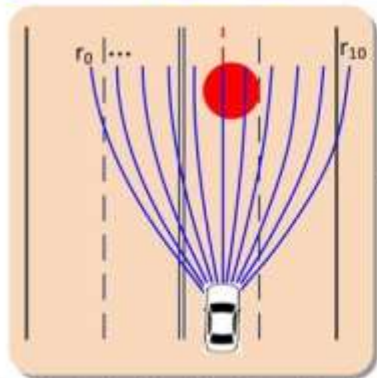


### Path Candidate Generation

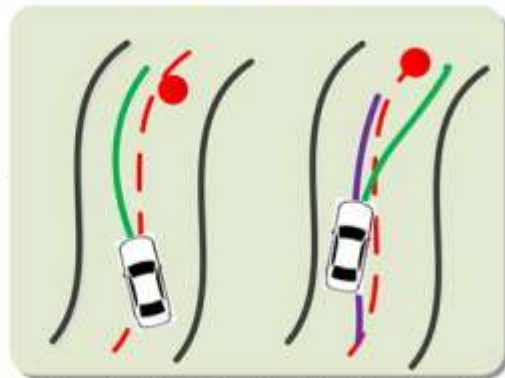


### Path Selection

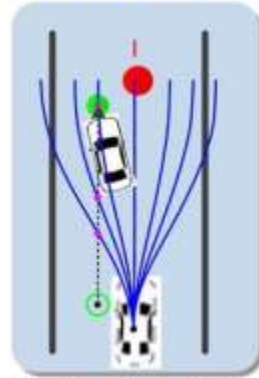
$$f = \omega_s f_s(r_i) + \omega_c f_c(r_i) + \omega_d f_d(r_i, a(r_i))$$



+

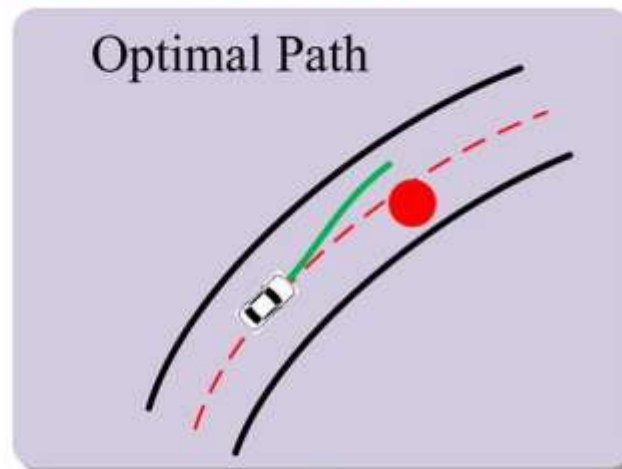


+

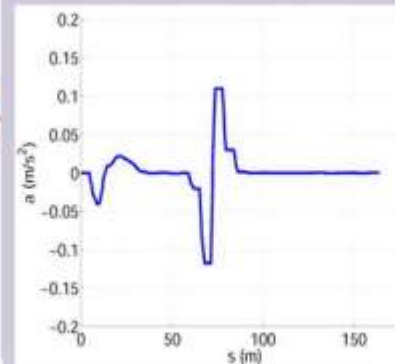


### Path Planning Results

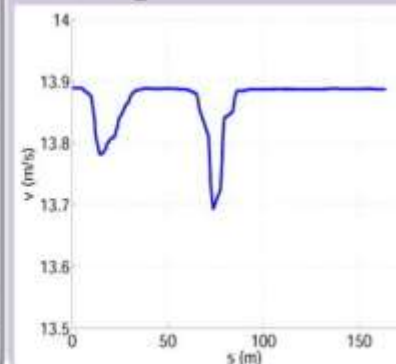
#### Optimal Path



#### Appropriate Acceleration



#### Appropriate Speed



	Common AI Techniques					
Use Case Families	Generative Models	Non-Generative ML	Optimisation	Simulation	Rules	Graphs
Forecasting	Low	High	Low	High	Medium	Low
Planning	Low	Low	High	Medium	Medium	High
Decision Intelligence	Low	Medium	High	High	High	Medium
Autonomous System	Low	Medium	High	Medium	Medium	Low
Segmentation	Medium	High	Low	Low	High	High
Recommender	Medium	High	Medium	Low	Medium	High
Perception	Medium	High	Low	Low	Low	Low
Intelligent Automation	Medium	High	Low	Low	High	Medium
Anomaly Detection	Medium	High	Low	Medium	Medium	High
Content Generation	High	Low	Low	High	Low	Low
Chatbots	High	High	Low	Low	Medium	High
Knowledge Discovery	High	Medium	Low	Low	Medium	High

# Iseõitvad autod on keerulisemad, kui algul paistis

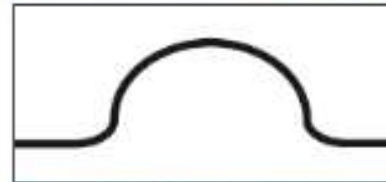
Chris Urmson, the former leader of Google's self-driving car project, once hoped that his son wouldn't need a driver's license because driverless cars would be so plentiful by 2020.

Now the CEO of the self-driving startup Aurora, Urmson says that driverless cars **„will be slowly integrated onto our roads over the next 30 to 50 years“**.



*Näide:*

Riiete triikimist ei suudeta  
praegu veel automatiseerida



*Näide:*

Laualt mustade nõude  
viimine nõudepesumasinasse  
ei ole nähtavas tulevikus veel  
automatiseeritav

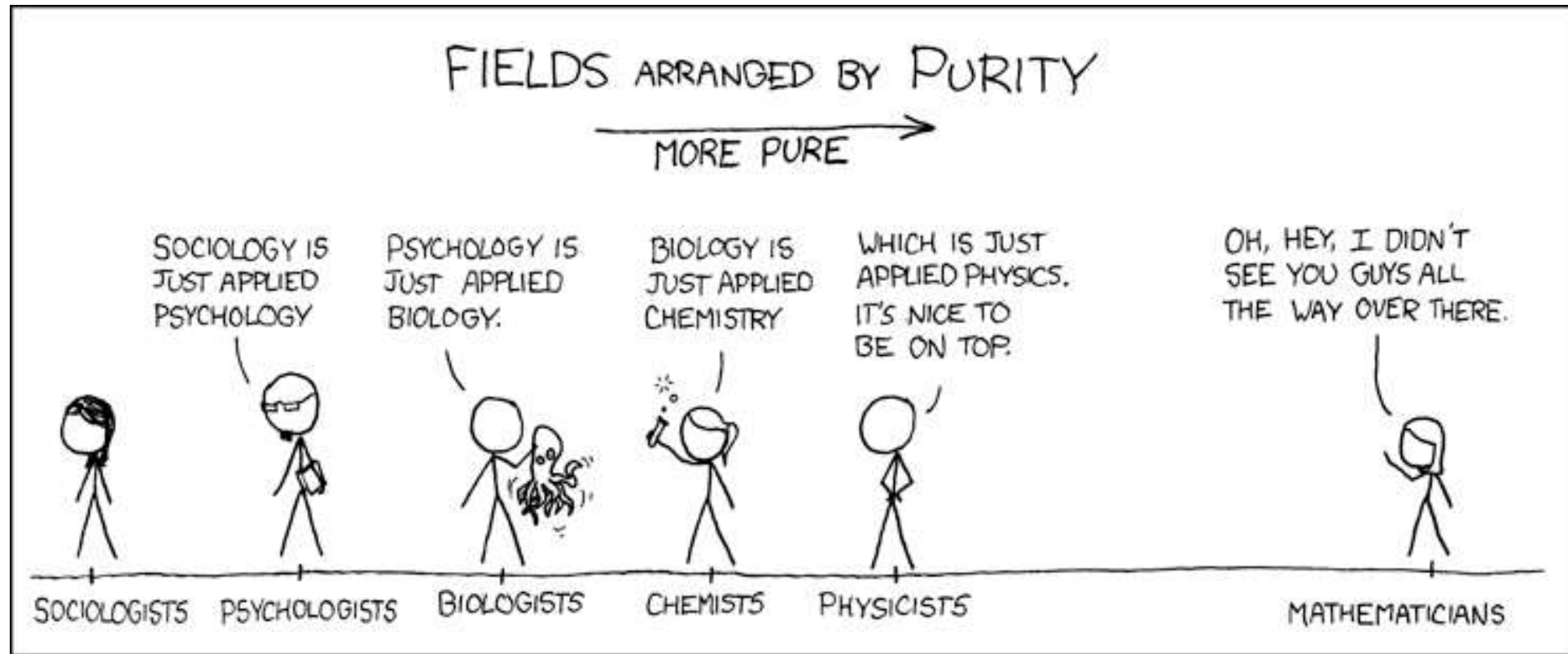


# Rodney Brooksi tähelepanekuid

- Kui A.I. saab mõnda tüüpi ülesandega hakkama, siis inimesed tüüpiliselt ülehindavad tema võimekust lähedase või seotud ülesande jaoks.
- Enamus edukaid A.I. rakendusi on süsteemides, kus nad toetavad inimehtöotajat ja ei saa ilma inimehtöotajata ise hakkama.
- Enamike edukate A.I. rakenduste jaoks on väga suur hulk erijuhtumeid, millega sama süsteem hakkama ei saa, ning millele sama süsteemi lihtsalt laiendada ei ole võimalik.



*Teoreetilisest arvutiteadusest*



## *Fundamentaalsed teoreetilise arvutiteaduse poolt*

Selgub, et iga täpselt formuleeritud probleemi (matemaatika- ja programmeerimisprobleemid) jaoks ei leidugi lahendavat algoritmi!

Näiteks, **peatumisprobleemi** jaoks ei ole lahendavat algoritmi.



## *Mis on peatumisprobleem?*

Osad programmid peatuvad, osad mitte. Näiteks:

```
def iamhalting(i): # halts for any i
    while i<10:
        print i*i
        i=i+1
```

```
def iamnothalting(i): # does not halt if i<=5
    while i<10:
        print i*i
        if i>5: i=i+1
```

Küsimus: kas saab kirjutada programmi, mis saaks sisendiks programmi **P** ja teksti **T** ja tuvastaks, kas programm sellele tekstile rakendamisel **P(T)** peatub?

## *Peatumisprobleemi võimatuse tõestus: 1. osa*

Oletame, et saab kirjutada peatumiskontrolli programmi

```
HaltAnswerer(prog, inputs) :  
    if halts (prog(inputs)) :  
        return true  
    else:  
        return false
```

## *Peatumisprobleemi võimatuse tõestus: 2. osa*

Kirjutame nüüd uue programmi `nasty`, mis kasutab `HaltAnswereri`

```
Nasty(prog) :  
    if HaltAnswerer(prog,prog)==true:  
        # case 1: Nasty will go into an infinite loop  
        while true: x=1  
    else:  
        # case 2: Nasty will halt  
        return
```

Kas `nasty(nasty)` peatub? Me ei tea, aga on ainult kaks võimalikku varianti:

- Kui **ei peatu**, siis: `HaltAnswerer(nasty,nasty)` annab “false”, seega [case 2] tõttu `nasty(nasty)` peatub. Vastuolu oletusega, et ei peatu!
- Kui **peatub**, siis: `HaltAnswerer(nasty,nasty)` annab “true”, seega [case 1] tõttu `nasty(nasty)` ei peatu. Vastuolu oletusega, et peatub!